In the Claims

1. (Currently amended) A method of ~~automatic configuration of~~ designing a microprocessor architecture whereby:

~~(a)~~ the architecture includes a number of execution units;

~~(b)~~ the architecture has configurable direct connectivity between ~~those~~ said execution units; and

~~(c)~~ the execution units are able to communicate data directly without the need to be connected between intermediate register files that are shared between ~~multiple~~ said execution units; and whereby:

~~(d)~~ ~~the~~ data and control flows within a particular input program are automatically analyzed during the design of the microprocessor architecture and results of the analysis are used to create a hardware description of the microprocessor architecture that is synthesizable into a physical hardware embodiment of said microprocessor architecture ~~influence decisions regarding execution unit replication and connectivity~~.

2. (Currently amended) The method according to claim 1 whereby the step of creating a hardware description of the microprocessor architecture includes the generation of one or more ~~multiple~~ candidate architectures ~~are generated~~.

3. (Currently amended) The method according to claim 2 whereby ~~the best~~ a final microprocessor architecture is automatically selected from said one or more candidate architectures ~~is automatically selected~~ on the basis of user defined metrics, thereby completing the design of said microprocessor architecture.

4. (Currently amended) The method according to claim 2 whereby data is output to allow the construction of a graph representing ~~the~~ characteristics of certain of said one or more candidate architectures ~~candidates~~.

5. (Currently amended) The method according to claim 2 whereby said hardware description of the microprocessor architecture includes descriptions of connections between and numbers of execution units comprising said microprocessor architecture, and whereby a number of new connections and or execution units are added to the hardware description of said microprocessor architecture on each generation of a candidate architecture.

6. (original) The method according to claim 5 whereby mapping of code onto a trial architecture is used to influence connectivity choices.

7. (Currently amended) The method according to claim 6 whereby the delays caused by execution unit conflicts ~~in the schedule are used to~~ increase the chances of an additional execution unit ~~of that type~~ being added to the microprocessor architecture.

8. (Currently amended) The method according to claim 1 whereby every candidate architecture generated contains a certain minimum set of execution unit types.

9. (original) The method according to claim 8 whereby there is a certain minimum connectivity between all execution units.

10. (Currently amended) The method according to claim 9 whereby the minimum connectivity guarantees that arbitrary new code sequences can be mapped to the microprocessor architecture.

11. (Currently amended) The method according to claim 10 whereby the guarantee of mapping of new code is performed using a reachability analysis between results and operands within the microprocessor architecture.

12. (Currently amended) The method according to claim 11 whereby the reachability analysis ensures that ~~the~~ result output from ~~of~~ every type of execution unit can be transported to the operand of every type of execution unit.

13. (original) The method according to claim 11 whereby the reachability analysis ensures that every result can be written to a central register file and that every operand can be read from a central register file.

14. (Currently amended) The method according to claim 1 whereby ~~the~~ an initial connectivity within the architecture is determined from data flows within graph representations of the input program.

15. (Currently amended) The method according to claim 5 whereby the new connections are added as a result of connections requested during the ~~code~~ candidate-architecture generation process.

16. (Currently amended) The method according to claim 15 whereby the addition of new connections is constrained by certain connectivity rules.

17. (Currently amended) The method according to claim 16 whereby the connectivity rules relate to maximum ~~number~~ numbers of operand inputs, maximum ~~number~~ numbers of result outputs and ~~the~~ an estimated connectivity distance.

18. (Currently amended) The method according to claim 17 whereby a set of potential new connections are maintained and those which are added are those which will improve ~~the~~ a fitness metric most but are within the constraints.

19. (original) The method according to claim 1 whereby the execution units are placed in a logical grid layout.

20. (original) The method according to claim 19 whereby the connectivity rules include a maximum distance between execution unit grid positions.

21. (original) The method according to claim 19 whereby the execution units are obtained from a component library, each of which includes a reference to the hardware description of the execution unit.

22. (Currently amended) The method according to claim 21 whereby the execution unit components have pre-characterised characteristics that ~~may~~ include one or more of area, maximum operational frequency and average power consumption.

23. (original) The method according to claim 22 whereby each architecture is designed to have a minimum operational frequency on a given implementation technology.

24. (original) The method according to claim 19 whereby the placement of execution units is initiated with the register file at the centre of the grid.

25. (original) The method according to claim 24 whereby the placement of execution units is performed from the centre outward in order of decreasing usage frequency of the execution units.

26. (Currently amended) The method according to claim 1 whereby the architecture ~~may be~~ <u>is</u> optimised for a certain set of functions that are specified to the system.

27. (original) The method according to claim 1 whereby the number and type of execution units and their connectivity for a given architecture is stored in a description file.

28. (Currently amended) The method according to claim 27 whereby the stored ~~information~~ <u>number and type of execution units and their connectivity for a given architecture</u> includes details of the execution word layout used to control each of the execution units.

29. (Currently amended) The method according to claim 28 whereby the stored ~~information~~ <u>number and type of execution units and their connectivity for a given architecture</u> includes details of ~~the~~ selection codes associated with operand inputs, output registers and execution unit operation codes.

30. (Currently amended) The method according to claim 27 whereby a top level hardware description language file ~~may be~~ <u>is</u> generated from the ~~information~~ stored <u>number and type of execution units and their connectivity for a given architecture</u> that instantiates connectivity and required execution units.

31. (Currently amended) The method according to claim 30 whereby ~~the~~ required package or library files are automatically generated to incorporate the hardware descriptions for all the required execution models for the purposes of hardware synthesis.

32. (Currently amended) The method according to claim 31 whereby instruction level profiling information is used to influence ~~the~~ weighting of individual instructions.

33. (Currently amended) The method according to claim 32 whereby the weighting of individual instructions during scheduling is related to ~~the~~ profile weighting.

34. (Previously presented) A microprocessor automatically configured using a method as defined in Claim 1.

35. (New) A method of automatic configuration of a microprocessor architecture whereby:

(a) the architecture includes a configurable number of execution units;

(b) the architecture has configurable connectivity between those execution units;

(c) the execution units are able to communicate data directly without the need to be connected between register files that are shared between multiple execution units; and

(d) the data and control flows within a particular input program are used to influence decisions regarding execution unit replication and connectivity,

whereby multiple candidate architectures are generated;

whereby a number of new connections and or execution units are added to the architecture on each generation;

whereby mapping of code onto a trial architecture is used to influence connectivity choices; and

whereby the delays caused by execution unit conflicts in the schedule are used to increase the chances of an additional execution unit of that type being added to the architecture.

36. (New) A method of automatic configuration of a microprocessor architecture whereby:

(a) the architecture includes a configurable number of execution units;

(b) the architecture has configurable connectivity between those execution units;

(c) the execution units are able to communicate data directly without the need to be connected between register files that are shared between multiple execution units; and

(d) the data and control flows within a particular input program are used to influence decisions regarding execution unit replication and connectivity;

whereby the execution units are placed in a logical grid layout; and

whereby the placement of execution units is initiated with the register file at the centre of the grid.